Network Working Group
Request for Comments: 3828
Category: Standards Track

L-A. Larzon
Lulea University of Technology
M. Degermark
S. Pink
The University of Arizona
L-E. Jonsson, Ed.
Ericsson
G. Fairhurst, Ed.
University of Aberdeen

July 2004

The Lightweight User Datagram Protocol (UDP-Lite)

Status of this Memo

Copyright Notice

Abstract

This document describes the Lightweight User Datagram Protocol (UDP-Lite), which is similar to the User Datagram Protocol (UDP) (RFC 768), but can also serve applications in error-prone network environments that prefer to have partially damaged payloads delivered rather than discarded. If this feature is not used, UDP-Lite is semantically identical to UDP.

Table of Contents

1. Introduction

This document describes a new transport protocol, UDP-Lite, (also known as UDPLite). This new protocol is based on three observations:

First, there is a class of applications that benefit from having damaged data delivered rather than discarded by the network. A number of codecs for voice and video fall into this class (e.g., the AMR speech codec [RFC-3267], the Internet Low Bit Rate Codec [ILBRC], and error resilient H.263+ [ITU-H.263], H.264 [ITU-H.264; H.264], and MPEG-4 [ISO-14496] video codecs). These codecs may be designed to cope better with errors in the payload than with loss of entire packets.

Second, all links that support IP transmission should use a strong link layer integrity check (e.g., CRC-32 [RFC-3819]), and this MUST be used by default for IP traffic. When the under-lying link supports it, certain types of traffic (e.g., UDP-Lite) may benefit from a different link behavior that permits partially damaged IP packets to be forwarded when requested [RFC-3819]. Several radio technologies (e.g., [3GPP]) support this link behavior when operating at a point where cost and delay are sufficiently low. If error-prone links are aware of the error sensitive portion of a packet, it is also possible for the physical link to provide greater protection to reduce the probability of corruption of these error sensitive bytes (e.g., the use of unequal Forward Error Correction).

Third, intermediate layers (i.e., IP and the transport layer protocols) should not prevent error-tolerant applications from running well in the presence of such links. IP is not a problem in this regard, since the IP header has no checksum that covers the IP payload. The generally available transport protocol best suited for these applications is UDP, since it has no overhead for retransmission of erroneous packets, in-order delivery, or error correction. In IPv4 [RFC-791], the UDP checksum covers either the entire packet or nothing at all. In IPv6 [RFC-2460], the UDP checksum is mandatory and must not be disabled. The IPv6 header does not have a header checksum and it was deemed necessary to always protect the IP addressing information by making the UDP checksum mandatory.

A transport protocol is needed that conforms to the properties of link layers and applications described above [LDP99]. The error-detection mechanism of the transport layer must be able to protect vital information such as headers, but also to optionally ignore errors best dealt with by the application. The set of octets to be verified by the checksum is best specified by the sending application.

UDP-Lite provides a checksum with an optional partial coverage. When using this option, a packet is divided into a sensitive part (covered by the checksum) and an insensitive part (not covered by the checksum). Errors in the insensitive part will not cause the packet to be discarded by the transport layer at the receiving end host. When the checksum covers the entire packet, which should be the default, UDP-Lite is semantically identical to UDP.

Compared to UDP, the UDP-Lite partial checksum provides extra flexibility for applications that want to define the payload as partially insensitive to bit errors.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC-2119].

3. Protocol Description

The UDP-Lite header is shown in figure 1. Its format differs from UDP in that the Length field has been replaced with a Checksum Coverage field. This can be done since information about UDP packet length can be provided by the IP module in the same manner as for TCP [RFC-793].

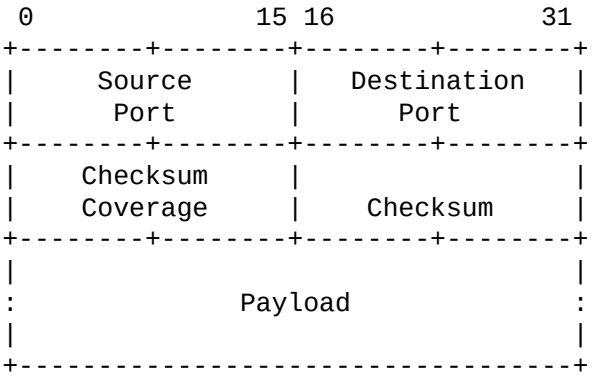

Figure 1: UDP-Lite Header Format

3.1. Fields

The fields Source Port and Destination Port are defined as in the UDP specification [RFC-768]. UDP-Lite uses the same set of port number values assigned by the IANA for use by UDP.

Checksum Coverage is the number of octets, counting from the first octet of the UDP-Lite header, that are covered by the checksum. The UDP-Lite header MUST always be covered by the checksum. Despite this requirement, the Checksum Coverage is expressed in octets from the beginning of the UDP-Lite header in the same way as for UDP. A Checksum Coverage of zero indicates that the entire UDP-Lite packet is covered by the checksum. This means that the value of the Checksum Coverage field MUST be either 0 or at least 8. A UDP-Lite packet with a Checksum Coverage value of 1 to 7 MUST be discarded by the receiver. Irrespective of the Checksum Coverage, the computed Checksum field MUST include a pseudo-header, based on the IP header (see below). UDP-Lite packets with a Checksum Coverage greater than the IP length MUST also be discarded.

The Checksum field is the 16-bit one’s complement of the one’s complement sum of a pseudo-header of information collected from the IP header, the number of octets specified by the Checksum Coverage (starting at the first octet in the UDP-Lite header), virtually padded with a zero octet at the end (if necessary) to make a multiple of two octets [RFC-1071]. Prior to computation, the checksum field MUST be set to zero. If the computed checksum is 0, it is transmitted as all ones (the equivalent in one’s complement arithmetic).

Since the transmitted checksum MUST NOT be all zeroes, an application using UDP-Lite that wishes to have no protection of the packet payload should use a Checksum Coverage value of 8. This differs

from the use of UDP over IPv4 in that the minimal UDP-Lite checksum always covers the UDP-Lite protocol header, which includes the Checksum Coverage field.

3.2. Pseudo Header

UDP and UDP-Lite use the same conceptually prefixed pseudo header from the IP layer for the checksum. This pseudo header is different for IPv4 and IPv6. The pseudo header of UDP-Lite is different from the pseudo header of UDP in one way: The value of the Length field of the pseudo header is not taken from the UDP-Lite header, but rather from information provided by the IP module. This computation is done in the same manner as for TCP [RFC-793], and implies that the Length field of the pseudo header includes the UDP-Lite header and all subsequent octets in the IP payload.

3.3. Application Interface

An application interface should allow the same operations as for UDP. In addition to this, it should provide a way for the sending application to pass the Checksum Coverage value to the UDP-Lite module. There should also be a way to pass the Checksum Coverage value to the receiving application, or at least let the receiving application block delivery of packets with coverage values less than a value provided by the application.

It is RECOMMENDED that the default behavior of UDP-Lite be set to mimic UDP by having the Checksum Coverage field match the length of the UDP-Lite packet and verify the entire packet. Applications that wish to define the payload as partially insensitive to bit errors (e.g., error tolerant codecs using RTP [RFC-3550]) should do this by an explicit system call on the sender side. Applications that wish to receive payloads that were only partially covered by a checksum should inform the receiving system by an explicit system call.

The characteristics of the links forming an Internet path may vary greatly. It is therefore difficult to make assumptions about the level or patterns of errors that may occur in the corruption insensitive part of the UDP-Lite payload. Applications that use UDP-Lite should not make any assumptions regarding the correctness of the received data beyond the position indicated by the Checksum Coverage field, and should, if necessary, introduce their own appropriate validity checks.

3.4. IP Interface

As for UDP, the IP module must provide the pseudo header to the UDP-Lite protocol module (known as the UDPLite module). The UDP-Lite pseudo header contains the IP addresses and protocol fields of the IP header, and also the length of the IP payload, which is derived from the Length field in the IP header.

The sender IP module MUST NOT pad the IP payload with extra octets, since the length of the UDP-Lite payload delivered to the receiver depends on the length of the IP payload.

3.5. Jumbograms

The Checksum Coverage field is 16 bits and can represent a Checksum Coverage value of up to 65535 octets. This allows arbitrary checksum coverage for IP packets, unless they are Jumbograms. For Jumbograms, the checksum can cover either the entire payload (when the Checksum Coverage field has the value zero), or else at most the initial 65535 octets of the UDP-Lite packet.

4. Lower Layer Considerations

Since UDP-Lite can deliver packets with damaged payloads to an application that wishes to receive them, frames carrying UDP-Lite packets need not be discarded by lower layer protocols when there are errors only in the insensitive part. For a link that supports partial error detection, the Checksum Coverage field in the UDP-Lite header MAY be used as a hint of where errors do not need to be detected. Lower layers MUST use a strong error detection mechanism [RFC-3819] to detect at least errors that occur in the sensitive part of the packet, and discard damaged packets. The sensitive part consists of the octets between the first octet of the IP header and the last octet identified by the Checksum Coverage field. The sensitive part would thus be treated in exactly the same way as for a UDP packet.

Link layers that do not support partial error detection suitable for UDP-Lite, as described above, MUST detect errors in the entire UDP-Lite packet, and MUST discard damaged packets [RFC-3819]. The whole UDP-Lite packet is thus treated in exactly the same way as a UDP packet.

It should be noted that UDP-Lite would only make a difference to an application if partial error detection, based on the partial checksum feature of UDP-Lite, is implemented also by link layers, as discussed above. Partial error detection at the link layer would only make a difference when implemented over error-prone links.

5. Compatibility with UDP

UDP and UDP-Lite have similar syntax and semantics. Applications designed for UDP may therefore use UDP-Lite instead, and will by default receive the same full packet coverage. The similarities also ease implementation of UDP-Lite, since only minor modifications are needed to an existing UDP implementation.

UDP-Lite has been allocated a separate IP protocol identifier, 136 (UDPLite), that allows a receiver to identify whether UDP or UDP-Lite is used. A destination end host that is unaware of UDP-Lite will, in general, return an ICMP "Protocol Unreachable" or an ICMPv6 "Payload Type Unknown" error message (depending on the IP protocol type). This simple method of detecting UDP-Lite unaware systems is the primary benefit of having separate protocol identifiers.

The remainder of this section provides the rationale for allocating a separate IP protocol identifier for UDP-Lite, rather than sharing the IP protocol identifier with UDP.

There are no known interoperability problems between UDP and UDP-Lite if they were to share the protocol identifier with UDP. Specifically, there is no case where a potentially problematic packet is delivered to an unsuspecting application; a UDP-Lite payload with partial checksum coverage cannot be delivered to UDP applications, and UDP packets that only partially fill the IP payload cannot be delivered to applications using UDP-Lite.

However, if the protocol identifier were to have been shared between UDP and UDP-Lite, and a UDP-Lite implementation was to send a UDP-Lite packet using a partial checksum to a UDP implementation, the UDP implementation would silently discard the packet, because a mismatching pseudo header would cause the UDP checksum to fail. Neither the sending nor the receiving application would be notified. Potential solutions to this could have been:

1) explicit application in-band signaling (while not using the partial checksum coverage option) to enable the sender to learn whether the receiver is UDP-Lite enabled or not, or

2) use of out-of-band signaling such as H.323, SIP, or RTCP to convey whether the receiver is UDP-Lite enabled.

Since UDP-Lite has been assigned its own IP protocol identifier, there is no need to consider this possibility of delivery of a UDP-Lite packet to an unsuspecting UDP port.

6. Security Considerations

The security impact of UDP-Lite is related to its interaction with authentication and encryption mechanisms. When the partial checksum option of UDP-Lite is enabled, the insensitive portion of a packet may change in transit. This is contrary to the idea behind most authentication mechanisms: authentication succeeds if the packet has not changed in transit. Unless authentication mechanisms that operate only on the sensitive part of packets are developed and used, authentication will always fail for UDP-Lite packets where the insensitive part has been damaged.

The IPsec integrity check (Encapsulation Security Protocol, ESP [RFC-2406], or Authentication Header, AH [RFC-2402]) is applied (at least) to the entire IP packet payload. Corruption of any bit within the protected area will then result in the IP receiver discarding the UDP-Lite packet.

When IPsec is used with ESP payload encryption, a link can not determine the specific transport protocol of a packet being forwarded by inspecting the IP packet payload. In this case, the link MUST provide a standard integrity check covering the entire IP packet and payload. UDP-Lite provides no benefit in this case.

Encryption (e.g., at the transport or application levels) may be used. If a few bits of an encrypted packet are damaged, the decryption transform will typically spread errors so that the packet becomes too damaged to be of use. Many encryption transforms today exhibit this behavior. There exist encryption transforms, and stream ciphers, which do not cause error propagation. Note that omitting an integrity check can, under certain circumstances, compromise confidentiality [Bellovin98]. Proper use of stream ciphers poses its own challenges [BB01]. In particular, an attacker can cause predictable changes to the ultimate plaintext, even without being able to decrypt the ciphertext.

7. IANA Considerations

A new IP protocol number, 136 has been assigned for UDP-Lite. The name associated with this protocol number is "UDPLite". This ensures compatibility across a wide range of platforms, since on some platforms the "-" character may not form part of a protocol entity name.

8. References

8.1. Normative References

[RFC-768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC-791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC-793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC-1071] Braden, R., Borman, D. and C. Partridge, "Computing the Internet Checksum", RFC 1071, September 1988.

[RFC-2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC-2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

8.2. Informative References

[Bellovin98] Bellovin, S.M., "Cryptography and the Internet", in Proceedings of CRYPTO '98, August 1988.

[BB01] Bellovin, S. and M. Blaze, "Cryptographic Modes of Operation for the Internet", Second NIST Workshop on Modes of Operation, August 2001.

[3GPP] "Technical Specification Group Services and System Aspects; Quality of Service (QoS) concept and architecture", TS 23.107 V5.9.0, Technical Specification 3rd Generation Partnership Project, June 2003.

[H.264] Hannuksela, M.M., Stockhammer, T., Westerlund, M. and D. Singer, "RTP payload Format for H.264 Video", Internet Draft, Work in Progress, March 2003.

[ILBRC] S.V. Andersen, et. al., "Internet Low Bit Rate Codec", Work in Progress, March 2003.

[ISO-14496] ISO/IEC International Standard 1446 (MPEG-4), "Information Technology Coding of Audio-Visual Objects", January 2000.

[ITU-H.263] "Video Coding for Low Bit Rate Communication," ITU-T Recommendation H.263, January 1998.

[ITU-H.264] "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification", ITU-T Recommendation H.264, May 2003.

[RFC-3819] Karn, Ed., P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J. and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.

[RFC-3550] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003.

[RFC-2402] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[RFC-2406] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[RFC-3267] Sjoberg, J., Westerlund, M., Lakeaniemi, A. and Q. Xie, "Real-Time Transport Protocol (RTP) Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs", RFC 3267, June 2002.

[LDP99] Larzon, L-A., Degermark, M. and S. Pink, "UDP Lite for Real-Time Multimedia Applications", Proceedings of the IEEE International Conference of Communications (ICC), 1999.

9. Acknowledgements

Thanks to Ghyslain Pelletier for significant technical and editorial comments. Thanks also to Steven Bellovin, Elisabetta Carrara, and Mats Naslund for reviewing the security considerations chapter, and to Peter Eriksson for a language review, thereby improving the clarity of this document.

10. Authors' Addresses

Lars-Ake Larzon
Department of CS & EE
Lulea University of Technology
S-971 87 Lulea, Sweden

EMail: lln@csee.ltu.se

Mikael Degermark
Department of Computer Science
The University of Arizona
P.O. Box 210077
Tucson, AZ 85721-0077, USA

EMail: micke@cs.arizona.edu

Stephen Pink
The University of Arizona
P.O. Box 210077
Tucson, AZ 85721-0077, USA

EMail: steve@cs.arizona.edu

Lars-Erik Jonsson
Ericsson AB
Box 920
S-971 28 Lulea, Sweden

EMail: lars-erik.jonsson@ericsson.com

Godred Fairhurst
Department of Engineering
University of Aberdeen
Aberdeen, AB24 3UE, UK

EMail: gorry@erg.abdn.ac.uk

11. Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.